Lab1 – Part 2

Adding the OpenGL Panel
Roger Crawfis

Create a PIXELFORMATDESCRIPTOR

- PIXELFORMATDESCRIPTOR * defaultFormat;
- defaultFormat = new PIXELFORMATDESCRIPTOR();
- defaultFormat->nSize = sizeof(PIXELFORMATDESCRIPTOR); // size of pfd
- defaultFormat->nVersion = 1; // version number
- defaultFormat->dwFlags = PFD_DRAW_TO_WINDOW | // support window
  PFD_SUPPORT_OPENGL | // support OpenGL
  PFD_DOUBLEBUFFER; // double buffered
- defaultFormat->iPixelType = PFD_TYPE_RGBA; // RGBA type
- defaultFormat->cDepthBits = 32; // 32-bit color depth
- defaultFormat->cColorBits = 24; // Select Our Color Depth
- defaultFormat->cAlphaBits = 0; // No Alpha Buffer
- defaultFormat->cAccumBits = 0; // No Accumulation Buffer
- defaultFormat->cStencilBits = 8; // No Stencil Buffer
- defaultFormat->cAuxBuffers = 0; // No Auxiliary Buffer
- defaultFormat->iLayerType = PFD_MAIN_PLANE; // Main Drawing Layer

Add an OpenGLPanel member

- Add a private member variable to Form1
  - private: OhioState::OpenGLPanel * glPanel;
- Add a “using namespace OhioState;” to avoid fully scoping this each time.
- In the Form1 constructor:
  - glPanel = new OpenGLPanel(*defaultFormat);
  - glPanel->Dock = System::Windows::Forms::DockStyle::Fill;
  - panel2->Controls->Add( glPanel );

Referencing other libraries

- In .NET everything is contained in an assembly.
- This protects from header files being different versions than the libraries or dll’s.
- We need to add a reference to the OpenGLPanel.
  - On the Solutions panel, right-click and select Add Reference.
  - Browse to where the OpenGLPanel.dll resides and select it. (this may add a fully qualified path name creating headaches for the grader and machine migration).
Adding the OpenGL library

- Download the latest OpenGL header files.
- Include the main OpenGL library. You also need to include windows.h:
  - #include <windows.h>
  - #include <GL/gl.h>
- Add a reference to the library in the Project’s Properties->Linker->Input->Additional Dependencies field:
  - opengl32.lib

Add a Paint event callback

- This is where you will do the bulk of the lab. For now, let’s draw a blueQuad in the window.
- public: void DrawLines( const int frameNumber )
  - {
    glColor3f( 0.2f, 0.2f, 1.0f );
    glBegin( GL_QUADS );
    glVertex3f( 0.2f, 0.2f, 0.0f );
    glVertex3f( 0.8f, 0.2f, 0.0f );
    glVertex3f( 0.8f, 0.8f, 0.0f );
    glVertex3f( 0.2f, 0.8f, 0.0f );
    glEnd();
  }

Add a Resize event callback

- Map the entire window to 0 to one.
- void ResizeEvent( const int width, const int height )
  - {
    glViewport(0, 0, width, height);
    glMatrixMode( GL_PROJECTION_MATRIX );
    glLoadIdentity();
    glOrtho( 0.0, 1.0, 0.0, 1.0, -1.0, 1.0 );
  }

Handling the events

- Okay, everything should compile, but we have not told the program to call these new methods.
- Set the callbacks for the OpenGLPanel in the Form1 constructor:
  - glPanel->AddRenderCallback( new OhioState::OpenGLRenderCallback( this, DrawScene) );
  - glPanel->AddResizeCallback( new OhioState::OpenGLResizeCallback( this, ResizeEvent) );
**Build and Run**

- Well something happened, but nothing is being drawn.
- This is because our OpenGLPanel does not make any assumptions, and hence leaves all work for you to do.
- In particular, we asked for double buffering, but we never swap the buffers.
- Add a `glPanel->SwapBuffers()` call, preferably in a PostRender callback.

**Working OpenGL?**

- This actually compiles and runs on my machine.
- Try resizing the window to force some Paint events. You can also drag a small window across it like a calculator.
- I am surprised this works. Why? Because we never explicitly cleared the buffer between paint events. OpenGLPanel overrides the `OnPaintBackground` method, but does nothing.
- We should add a PreRender callback that does `glClear(GL_COLOR_BUFFER_BIT)`

**Lab1 Status**

- You should have something like this now.

![Lab1 Status Image]

**Lab1 Status**

- Okay, we now have a start on our GUI and document, we have an OpenGL canvas or panel that we can draw to, all we need is the logic.
- Part 3 will look at some of the elements we need for a working lab.